65.00 220

IN THE UNITED STATES

PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS

AND INTERFERENCES

APPLICATION OF)
)
MARTIN G. REIFFIN)
)
FILED: APRIL 3, 1985)
)
FOR : COMPUTER SYSTEM WITH)
REAL-TIME CODE PROCESSING)
)
SERIAL NO. 719,507)

ART UNIT 232

EXAMINER T. LEE

BRIEF FOR APPELLANT

This is an appeal from the Examiner's final rejection dated March 9, 1988 as modified by the advisory action dated May 23, 1988. The claims on appeal are Claims 51, 54 and 56.

INTRODUCTORY STATEMENT

The subject computer system provides real-time processing of textual source code concurrently as the code is being entered or edited at the computer keyboard. The processor of the specific embodiment disclosed for illustration is a compiler of a formal language such as used for writing computer programs. However the claims on appeal are not so limited and are intended to cover the processing of any textual code including, for example, a natural language.

Since humans write programs in a programming language and computers execute only machine language, it is frequently necessary to translate from one language to the other. When the programming language is "high level", that is, abstract in the sense that it does not explicitly manipulate the computer registers and other hardware, the translation of the original program is performed by another program called a "compiler". The original program is called the "source code", and the resulting program translation is called the "object code".

Programming in a high-level language, as conducted in accordance with the prior art, is slow, tedious and inefficient. It requires a repeated sequence of steps comprising loading the editor, writing or editing the source code, loading the compiler, executing the compiler, loading the linker, executing the linker, running the program, and repeating the sequence when an error is indicated during compilation of the source code or execution of the object code. During much of the time the programmer is compelled to wait for completion of the loading or execution steps, and this waiting time is both wasteful and boring. As a result, the programming process is slow and expensive. The writing and processing of text in a natural language involves similar analyses and presents similar problems.

THE PRESENT INVENTION

The present invention obviates these problems by processing the textual source code concurrently in real time as the code is being entered or edited at the keyboard. As a result processing of the text may be completed almost immediately after it has been entered or edited, and lexical, syntactic and semantic errors are promptly revealed by error messages concurrently during the entry and editing stages.

As shown in Fig. 3, the compiler normally has control of the CPU and is in the process of analysing and translating the source code in the buffer. The occurrence of a keystroke at the terminal causes an interrupt, whereupon the CPU is vectored to the interrupt service routine. The latter includes an editor which is then executed by the CPU. If the keystroke corresponds to a control character, then an editing procedure such as a cursor movement, screen scroll, character deletion, or line deletion is performed. If the keystroke corresponds to an alphanumeric character or other valid source code character the latter is entered into the source code buffer and displayed on the video screen, and the screen cursor is advanced to the next character position.

The RET instruction is finally executed so that control of the CPU then immediately and automatically returns to the compiler. The compiler then resumes execution from the point in the source code where it was interrupted.

In more detail, when the programmer strikes a key on the

console keyboard the central processing unit executes the following interrupt sequence described for an 8080 or Z80 microprocessor: Upon completion of the instruction currently being executed the central processing unit will enter the interrupt mode and communicate its new status to the system by emitting an interrupt acknowledge signal.

Upon receipt of this signal the interrupt hardware gates an RST instruction onto the data bus. The central processing unit then executes the RST instruction. The latter is a one-byte call to a selected location in low memory where there is stored a jump instruction ("vector") to the interrupt service routine comprising the editor.

The interrupt service routine first saves the stack pointer and other CPU registers. A character code corresponding to the struck key is placed into the source code buffer. The buffer pointer is then advanced to the next location, the CPU registers are restored, the CPU interrupt enabled, and the RET (return) instruction executed to return control to the compiler immediately and automatically. This sequence is repeated for each successive struck key as the programmer strikes additional keys at the keyboard, the successive characters being entered into successive locations in the source code buffer as the buffer pointer advances. Upon return to the compiler from the interrupt service routine the compiler performs its lexical, syntactic and semantic analyses on the source code stored in the buffer.

Referring to Fig. 2, there are shown the circuitry and hardware components directly involved in the interrupt operation. Upon striking a key of the CRT console a train of pulses constituting the byte corresponding to the struck key is emitted

from the RS-232-C serial port O. A converter gate C1 converts the pulse train from RS-232-C levels to TTL (transistor-transistor-logic) levels to match the requirements of input port IN of the UART. The latter forms the serial pulse train into an eight-bit byte which is stored in the received data register (not shown) of the UART. The latter then outputs a data available signal at pin DAV which signal is transmitted by gate G1 to a vectored interrupt input VI of the PRIORITY ENCODER. The latter transmits a three-bit code A0,A1,A2 to the respective inputs of the INTERRUPT VECTOR REGISTER. The other five inputs of the latter are held positive by potential source +V, so that the resulting byte input to this register chip constitutes an RST call instruction described below. The signal at output pin GS of the priority encoder is transmitted by gate G2 to the latch enable input LE of the interrupt vector register to cause the latter to latch the RST call instruction into its internal flip-flops.

Activation of output pin GS of the priority encoder also transmits an interrupt signal through AND gate A1 to the interrupt request pin INT* of the Z80 CPU. Assuming that the interrupt mode of the processor is enabled, upon completion of the present instruction the CPU's status pins IORQ* and M1* are activated and their signals are transmitted by gates G3,G4 to AND gate A2 to form the INTA (interrupt acknowledge) signal. The latter is inverted by gate G5 and fed to the output enable pin OE of the interrupt vector register whereupon the RST call instruction at the inputs of the latter is jammed onto the DATA BUS.

The RST instruction is then input to and executed by the

Z80 CPU, causing the latter to push the contents of the program counter onto the stack, and further causing the CPU to jump to a predetermined location in low memory. This location stores a "vector" or three-byte JMP (jump) instruction to an interrupt service routine stored in memory. The latter includes the editor as well as a subroutine to store the contents of the CPU registers. Control of the CPU is then retained by the editor until either a character has been entered into the source code buffer or an editing operation has been completed.

The editor includes an input instruction which when executed causes the CPU to place the address of the UART's port on the ADDRESS BUS. The byte in the received data register (not shown) of the UART is then gated onto the DATA BUS and transmitted to the accumulator within the CPU. This byte symbolizes the character corresponding to the struck key. The editor then reads this byte from the accumulator, stores it in the source code buffer and displays it on the console screen.

THE CLAIMS ON APPEAL

Claims 51, 54 and 56 are on appeal and are copied as follows:

51. A computer system for the concurrent entry and processing of code, and comprising

a console having keys for entry of successive characters constituting said code,

a memory storing a buffer for holding said code, a code processor program consisting of a sequence of machine instructions, an interrupt service routine at a predetermined memory address, and a stack,

a central processing unit for executing said code processor program instructions and having an interrupt input, a program counter for storing the memory address of the next instruction to be executed by the central processing unit, and means immediately responsive to activation of the interrupt input to push the contents of the program counter onto said stack and to load into said program counter said memory address of the interrupt service routine,

means to cause the central processing unit to execute said code processor program instructions whereby the code processor program thereafter continuously maintains control of the central processing unit, except during interrupts, so as to process the code concurrently as the code is being entered into the system,

means responsive to striking one of said keys to activate said interrupt input whereby the contents of the program counter are pushed onto the stack and the memory address of the interrupt service routine is loaded into the program counter thereby immediately passing control of the central processing unit to the interrrupt service routine, whereupon the code processor program is interrupted immediately upon completing the machine instruction which it was executing when the key was struck,

said interrupt service routine including an editor program for inserting into said buffer a character code corresponding to said struck key, and having a return instruction at the end thereof,

said central processing unit having means responsive to execution of said return instruction to pop from the stack into the program counter the memory address of the code processor program instruction immediately following the instruction interrupted by striking said key, whereby the code processor program continues processing said character codes at the location in the buffer where the code processor program was interrupted.

54. A computer system for the concurrent entry and processing of code, and comprising

a console having keys for entry of successive characters constituting said code,

a memory storing a buffer for holding said code, a code processor program, and an interrupt service routine,

a central processing unit for executing said code processor program and having an interrupt input, and means responsive to activation of the interrupt input to call said interrupt service routine,

means to cause the central processing unit to execute said code processor program instructions whereby the code processor program thereafter continuously maintains control of the central processing unit, except during interrupts, so as to process the code concurrently as the code is being entered into the system,

means responsive to striking one of said keys to activate said interrupt input to interrupt the code processor program and pass control of the central processing unit to the interrrupt service routine,

said interrupt service routine including an editor program for inserting into said buffer a character code corresponding to said struck key, and

means thereafter to return control of the central processing unit to the code processor program to enable the latter to resume processing said character codes at the location in the buffer where the code processor program was interrupted.

56. A computer system for the concurrent entry and processing of code, and comprising

a console having keys for entry of successive characters constituting said code,

a memory storing a buffer for holding said code, an editor program and a code processor program,

a central processing unit,

means to maintain control of the central processing unit by said code processor program except when a key is struck whereby the code processor program thereafter processes the code concurrently as the code is being entered into the system,

means responsive to striking one of said keys to pass control of the central processing unit to the editor program for inserting into said buffer a character code corresponding to said struck key, and

means thereafter to return return control of the central processing unit to the code processor program to enable the latter to resume processing the code.

THE REFERENCE

Claims 51, 54 and 56 were submitted after the final rejection of March 9, 1988 and received only an advisory action. The latter did not state the grounds for non-allowance. However the Examiner informed appellant by telephone on July 11, 1988 that these claims were deemed to be unpatentable over the Lawrence et al. patent No. 4,464,730 under 35 U.S.C. 103.

Lawrence et al. disclose in Fig. 5 a collection of rectangular boxes purporting to represent a text terminal comprising a keyboard 1 for entering text and connected to a keyboard adapter 30 interfacing with a microprocessor 26. The latter is described as connected "with various components" through a data bus 27, an address bus 28 and an interrupt line 29. A communication adapter 31 connects the microprocessor 31 to a host processor shown only as a legend.

A sectioned rectangle 33 is referred to as a random access memory comprising a section 38 which purportedly "contains code for controlling the segmenting, desegmenting, formatting and editing processes 5, 11, 6 and 10, Fig. 1."

The text processing system receives a stream of text from the host processor. The text is stored in text store 2 where it is read by an "interpreter/formatter" 6. The latter appears to be an undisclosed program stored in memory section 38 and is shown in Figs. 1 and 4 as a box 6 labeled "I/F", in Fig. 3 as a box 6 labeled "INTERPRETER/FORMATTER', and in Fig. 5 as a box section 38 labeled "(TEXT CODE)".

The interpreter/formatter 6 purportedly "maps the text

stream into a refresh buffer (RB) 8 from whch the CRT display can be periodically refreshed." A keystroke processor 9 shown in Fig. 1 receives keystrokes from keyboard 1 and invokes a text editor 10 which modifies the text in store 2. The modified text is subsequently formatted and mapped into refresh buffer 8 for display on the screen of CRT display 3.

The critical aspects of the Lawrence disclosure are those relating to the modes of interaction between the interpreter/formatter 6 and the text editor 10. These modes are described in the Lawrence et al. specification as follows:

(1) the interpreter/formatter 6 is "initiated...by the text editor 10 on completion of an update to the text in the store 2" (Col. 3, Lines 25-29);

(2) the interpreter/formatter 6 is first "invoked" by the text editor 10 "to determine the physical cursor position relative to the text in the store 2", and after an insertion or deletion is "re-invoked" by the text editor 10 "to display the effect of the change on the screen" (Col. 3, Lines 30-38); and

(3) the "interpreter" (sic) is "terminated at the end of the current row" if "a key is depressed during formatting" (Col. 11, Lines 51-59).

THE EXAMINER'S POSITION

The three claims on appeal were added after the final rejection. Because these claims are in some respects broader and in other respects narrower than the canceled claims, and further in view of the absence of any stated grounds of rejection in the advisory action, appellant is not aware of the Examiner's position relative to the appealed claims beyond the Examiner's telephone statement that he deemed them to be unpatentable over the Lawrence et al. reference under 35 U.S.C. 103. However, the Examiner's view of this reference relative to the canceled claims not on appeal, as stated in Paragraph 11 of the final rejection of March 9, 1988, may be useful and is quoted below without the original underlining:

"11 During the formatting operation, an interpreter/formatter normally has the control of the system. Upon the activation of a keystroke, the interpreter/formatter is interrupted and the control of the system is passed to the editor. After the editor has completed the entry of the code byte, the system control is returned to the interpreter/formatter. Col. 13 lines 23 - 29 and col. 14 lines 14 - 18 and lines 26 - 30 clearly defines the interaction between the editor and the interpreter/formatter. Specifically when the interpreter/formatter has control of the system in performing its functions, editing means includes means to 'interrupt said formatting means' upon receipt of a 'keystroke from said keyboard' invoking said editing means to 'perform a text editing operation', col. 14 lines 26 - 30. Thereafter, means in said editing means for 'invoking said formatting means' when a text editing step has been completed, col. 14 lines 15 - 17. Furthermore, the editing means performs its editing function on a keystroke-by-keystroke basis, col. 13 line 29. In other words, the control of the system is passed back to the interpreter/formatter from the editor after each of the keystroke [sic]. Also see col. 11 lines 51 - 59 and col. 12 lines 50 - 53. In addition, col. 7 lines 16 - 28 shows the priorities of the different system functions. Within level 3, the highest priority starts from keystroke processing, to text editing, to interpreting. This priority scheme reinforces the interaction between the interpreter/formatter and the editor discussed hereinabove."

The Examiner's view of the law relating to the

requirement that a reference disclosure be enabling is all important and probably determinative of this appeal. In Paragraph 10 of the Action mailed March 19, 1987 the Examiner stated:

"10. First of all, applicant raised the question in regarding of the sufficiency of the disclosure of the cited reference, Lawrence et al. Lawrence et al is cited for its "disclosed" teachings. Therefore, as far as the prosecution of the subject application is concerned it is immaterial whether the disclosure of Lawrence et al meets the requirement of the 35 U.S.C. 112 first paragraph so long as the teaching relied upon by the examiner is adequately disclosed therein." (Emphasis added.)

APPELLANT'S CONTENTIONS

Appellant contends that:

I. The Lawrence et al. disclosure of the alleged "interactions" attributed by the Examiner to the interpreter/formatter 6 and text editor 10 is not "enabling" so as to constitute a valid reference. These alleged interactions are stated in Paragraph 11 of the final rejection of March 9, 1988 as follows:

"During the formatting operation, the interpreter/formatter normally has control of the system."

"Upon the activation of a keystroke, the interpreter/formatter is interrupted and the control of the system is passed to the editor."

"After the editor has completed the entry of the code byte, the system control is returned to the interpreter/formatter."

"In other words, control of the system is passed back to the interpreter/formatter from the editor after each of the keystroke [sic]."

It will be urged below that these attributed interactions are not enabled because the Lawrence et al. patent does not

contain an enabling disclosure of either their interpreter/formatter 6 or their text editor 10. These components are "disclosed" in the drawings only as labeled boxes and in the specification only as goals or objects to be achieved. Appellant contends that a box in the drawings labeled "INTERPRETER/FORMATTER" or "TEXT CODE" does not teach one of ordinary skill how to write software or build hardware, nor does a specification which summarily states desired interactions and merely lists the names of functions, goals and objects said to be achieved by unidentified and undisclosed programs.

II. A critical portion of the Lawrence patent (Column 14, Lines 15-17) expressly and repeatedly relied upon by the Examiner is a claim limitation added by an amendment filed by the patentees on February 17, 1984. This date is later than appellant's filing date of Sept. 28, 1982 of his parent application Serial No. 425,612. In Paragraph 11 of the Action mailed Sept. 18, 1987 the Examiner held that the present application is entitled to this earlier filing date which antedates the Lawrence et al. amendment relied upon by the Examiner.

III. The alleged interactions attributed by the Examiner to the Lawrence et al. interpreter/formatter 6 and text editor 10 are undisclosed in the patent, are inconsistent with the explicit verbiage of the patent specification, and are suggested only by appellant's own disclosure and claims.

IV. Appellant's Claims 51 and 54 recite other additional limitations neither discussed nor noted in the final rejection and which limitations patentably distinguish these two claims over the Lawrence et al. reference.

GOOD AND SUFFICIENT REASONS
WHY THE THIRD REIFFIN AFFIDAVIT
WAS NOT EARLIER PRESENTED

This is to present, pursuant to 37 C.F.R. 1.195, good and sufficient reasons why the accompanying affidavit entitled "THIRD AFFIDAVIT OF MARTIN G. REIFFIN" was not earlier presented.

The final rejection dated March 9, 1988 presented new grounds of rejection. This caused appellant to cancel all twenty-five (25) of the rejected claims and substitute six (6) new claims therefore, and to submit a copy of the third Wadsworth affidavit originally filed in appellant's copending parent application Serial No. 425,612.

Several of the new claims submitted after final rejection were substantially different from the previous claims and the reference in that they recited specifically and for the first time the structure and operation of appellant's interrupt mechanism. The Wadsworth affidavit attempted to explain the Lawrence et al. patent with respect to the issues of anticipation and non-enablement.

In the subsequent advisory action the Examiner indicated that three of the new claims were not allowable, and in a telephone inquiry by appellant the Examiner stated that these claims were deemed unpatentable over the Lawrence et al. patent under 35 U.S.C. 103.

At this point it became apparent to appellant that the Examiner was not familiar with the structure and operation of computer interrupts, and that the Wadsworth affidavit was at too

technical a level to explain successfully the structure and operation of the reference and its differences from the interrupt mode of operation claimed by appellant specifically for the first time in several of the new claims. It therefore occurred to appellant that a new affidavit, more explicit than the Wadsworth affidavit, would be necessary to provide an understanding of the complex interrupt technology involved in two of the disallowed claims and also involved in the issue of lack of an enabling disclosure in the reference.

It is respectfully submitted that an understanding of the technology and issues and a knowledgeable determination thereof requires a study of the facts and explanations set forth in the third Reiffin affidavit and that the affidavit should be admitted for this purpose.

A DISCLOSURE MUST BE ENABLING TO BE ANTICIPATORY

In Paragraph 10 of the Action mailed March 19, 1987 the Examiner stated that, "it is immaterial whether the disclosure of Lawrence et al meets the requirement of the 35 U.S.C. 112 first paragraph so long as the teaching relied upon by the examiner is adequately disclosed therein."

The Examiner apparently believes that a teaching which does not meet the enabling requirement of 35 U.S.C. 112 may nevertheless be "adequately disclosed" so as to constitute a valid anticipatory reference. This view is clearly contrary to the law, and correction of this error should be determinative so as to dispose of all issues in this appeal.

It is settled law that a prior patent or publication can be anticipatory only if its disclosure is enabling; that is, the disclosure must be sufficient to enable one of ordinary skill in the art to reduce the claimed invention to practice. The following decisions by the Court of Customs and Patent Appeals make this clear:

In In re LeGrice (CCPA 1962) 133 USPQ 365,373 the Court held:

" *** We think the board and the examiner were incorrect in overlooking these cases and interpreting the clause 'described in a printed publication' in section 102(b) according to what the examiner called 'its exact and unequivocal meaning.' When used in the Patent Act of 1952, this clause had acquired in the context in which it is used in section 102(b) something other than an 'exact and unequivocal meaning' as a result of the judicially imposed limitation that this clause requires that the description of the invention in the printed publication must be an 'enabling' description. Our study of the prior cases which have imposed this interpretation on the clause indicates that the proper test of a description in a publication as a bar to a patent as the clause is used in section 102(b) requires a determination of whether one skilled in the art to which the invention pertains could take the description of the invention in the printed publication and combine it with his own knowledge of the particular art and from this combination be put in possession of the invention on which a patent is sought. Unless this condition prevails, the description in the printed publication is inadequate as a statutory bar to patentability under section 102(b)." (Emphasis added.)

In In re Wilder (CCPA 1970) 166 USPQ 545,548 the Court held:

"Simply stated, a prior publication or patent description will be considered as anticipatory when its disclosure is at once specific and enabling with regard to the particular subject matter at issue. In effect, a prima facie case is made out whenever a reference is shown to contain a disclosure which is specific as to every critical element of the appealed claims. However, such disclosure may yet be held not to legally anticipate the claimed subject matter if it is found not to be sufficiently enabling, in other words, if it does not place the subject matter of the claims within 'the possession of the public.' See, e.g., In re LeGrice, 49 CCPA 1124, 301 F.2d 929, 133 USPQ 365 (1962); In re Brown, 51 CCPA 1254, 329 F.2d 1006, 141 USPQ 245 (1964)." (Emphasis added.)

In In re Wertheim and Mishkin (CCPA 1981) 209 USPQ 554,564-566 the Court held:

"Thus, the determinative question here is whether the invention claimed in the Pfluger patent finds a supporting disclosure in compliance with section 112, as required by section 120, in the 1961 Pfluger I application so as to entitle that invention in the Pfluger patent, as 'prior art,' to the filing date of Pfluger I. Without such support, the invention, and its accompanying disclosure, cannot be regarded as prior art as of that filing date."

* * *

"The board erred in ruling that since 'the substance of the relevant disclosure in Pfluger I was carried forward into the patent,' that same disclosure in the reference patent was entitled to the Pfluger I filing date, even though the entire patent was not. While some of the reference patent disclosure can be traced to Pfluger I, such portions of the original disclosure cannot be found 'carried over' for the purpose of awarding filing dates, unless that disclosure constituted a full, clear, concise and exact description in accordance with section 112, first paragraph, of the invention claimed in the reference patent, else the application could not have matured into a patent, within the Milburn section 102(e) rationale, to be 'prior art' under section 103."

* * *

"The dictum in Lund, supra, that

'*** the continuation-in-part application is entitled to the filing date of the parent application as to all subject matter carried over into it from the parent application *** for purposes of *** utilizing the patent disclosure as evidence to defeat another's right to a patent *** [emphasis in original]

is hereby modified to further include the requirement that the application, the filing date of which is needed to make a rejection, must disclose, pursuant to sections 120/112 the invention claimed in the reference patent." (Emphasis added.)

In In re Sheppard (CCPA 1964) 144 USPQ 42,45 the Court held:

"***The uncertainty which pervades the teaching of the reference leads us to conclude that Emeleus cannot properly be held an enabling disclosure."

* * *

"*** The Emeleus article is not so particular and definite that, without undue experimentation, one versed in the art to which it pertains could gain possession of the claimed subject matter. Under the circumstances we feel obliged to reverse the rejection of claims 2 and 3 under 35 U.S.C. 102(a)."

In In re Borst (CCPA 1965) 145 USPQ 554,557 the Court held:

"The mere fact that a disclosure is contained in a patent or application and thus 'constructively' reduced to practice, or that it is found in a printed publication, does not make the disclosure itself any more meaningful to those skilled in the art (and thus, ultimately, to the public). Rather, the criterion should be whether the disclosure is sufficient to enable one skilled in the art to reduce the disclosed invention to practice. In other words, the disclosure must be such as will give possession of the invention to the person of ordinary skill. Even the act of publication or the fiction of constructive reduction to practice will not suffice if the disclosure does not meet this standard. See In re Sheppard and In re LeGrice, supra."

Although it has been held that a component may be disclosed as a labeled box in the drawing if it is conventional and available, In re Donohue (CCPA) 193 USPQ 136, this is not applicable to the unconventional "interactions" relied upon by the Examiner. Neither appellant nor Lawrence et al. regards the interactions of their respective text processor and editor to be conventional since these respective interactions are claimed in both the reference patent and the present application. Furthermore, appellant contends that the alleged "interactions" attributed to the reference by the Examiner are so unconventional that they are not even disclosed in the reference. If the interactions relied upon by the Examiner were conventional then the Examiner could have readily cited a better reference having an enabling disclosure, and the Examiner would not have been compelled to rely upon a claim limitation added by amendment to the Lawrence application long after applicant's entitled filing date.

THE LAWRENCE DISCLOSURE IS NOT ENABLING WITH RESPECT TO THE ALLEGED INTERACTIONS

The issue of whether the Lawrence et al. disclosure is enabling with respect to the alleged interactions attributed to their interpreter/formatter 6 and text editor 10 by the Examiner is controlled by the decision of the Court of Appeals for the Federal Circuit in White Consolidated Industries, Inc. v. Vega Servo-Control, Inc. (CAFC 1983) 218 USPQ 961. The translator program disclosure held non-enabling by the Court in that case was far more instructive and meaningful than the Lawrence et al. translator program "disclosure" of a mere box 38 labeled "TEXT CODE".

The following quotations make clear the facts, issues and decision of this aspect of the White Consolidated case:

" ***

"The '653 system also includes a universal input feature so that a single part program can be used to control a plurality of machine tools, thus eliminating the need to create a new part program for each tool. This feature is accomplished by writing the part program in a universal NC language (i.e., machine tool independent) and employing a language translator in the control system to translate the program into machine code to control the tool. Describing the language translator, the '653 patent reads:

> The language TRANSLATOR used in the RUN mode may be a known translator capable of converting, in a single pass, a part program in programming language form into a part program in machine language form, as for example SPLIT (Sundstrand Program Language Internally Translated). In the CONVERSATIONAL mode, where each source or language part instruction is individually translated into machine language form, the TRANSLATOR program is modified by the additions shown in FIG. 12.

"At the time the application that resulted in the '653 patent was filed, SPLIT was a trade secret of Sunstrand, White's predecessor in interest, and was available only by purchase from Sunstrand.

"In holding the '653 patent invalid, Judge Cohn determined

that (1) the language translator was an integral part of the '653 system; (2) SPLIT was the only single pass language known to work in the '653 system at the time and was considered by the investors to be the best mode; and (3) by failing to disclose SPLIT, the '653 patent failed of compliance with the enablement and best mode requirements of 35 U.S.C. 112.

"Issues

"(1) Whether Judge Cohn erred in holding the '653 patent invalid for noncompliance with the enablement requirement of 35 U.S.C. 112.

***

"Opinion

"(1) Enablement under 35 U.S.C. 112

"[1] 35 U.S.C. 112 requires that the invention be described 'in such full, clear, concise, and exact terms as to enable any person skilled in the art *** to make and use the same.' White does not claim that SPLIT was disclosed, but rather that the specification contains an enabling disclosure notwithstanding its omission. White says the '653 patent calls for a known or standard single pass translator, as for example SPLIT, and specifies the characteristics of such a translator; that SPLIT was only an example; and that there were other known single pass translators interchangeable with SPLIT. White says because those other translators, e.g., ACTION and COMPACT, were known to those skilled in the art and available to them, the enablement requirement is satisfied.

"[2] We disagree. Though one may refer to an element of a claimed invention held as a trade secret by name only and yet satisfy 35 U.S.C. 112 if equivalent elements are known, and known to be equivalents, and available to those skilled in the art, In re Gebauer-Fuelnegg, et al., 50 USPQ 125, 121 F.2d 505 (CCPA 1941), there is insufficient evidence here from which to conclude that suitable substitutes for SPLIT were known and widely available. Testimony that ACTION and COMPACT were 'take-offs' of, i.e., patterned upon SPLIT, does not, for example, establish that those translators were known to be suitable substitutes for SPLIT. That other translators were available when the application was filed is unavailing where there is no basis in the record for finding that a person skilled in the art on reading the specification would know that another single pass processor would be suitable." (Emphasis added.)

The Court of Appeals thus held that where a named unavailable program is an "integral part" of the system the disclosure is enabling only if:

(1) "suitable substitutes ... were known and widely available"; and

(2) "a person skilled in the art on reading the specification would know that another ... processor would be suitable".

In the White Consolidated case the translator program SPLIT was identified in the specification, was in existence and was available by purchase from the patentee. In the present case the Lawrence phrase "interpreter/formatter" (deemed by the Examiner to be a "translator") does not identify any known available program. As alleged in Paragraph 8 of the THIRD AFFIDAVIT OF MARTIN G. REIFFIN the somewhat cryptic words "interpreter/formatter" appear to be a coined phrase which is not a term of art, and is not defined, explained or disclosed in any dictionary, treatise, encyclopedia, patent or other publication, Therefore this term is almost meaningless from the standpoint of enablement.

In the White Consolidated case there was evidence that other translators ACTION and COMPACT were available as suitable substitutes for SPLIT, although not so disclosed in the specification. In the present case there is no evidence that there is any available substitute for the Lawrence "translator" referred to as an "interpreter/formatter."

Therefore, if the disclosure of the translator program in the White Consolidated case is non-enabling, then the Lawrence et

al. "disclosure" of their "translator" program must be deemed even less so.

To summarize the issue, a box labeled "Text Code" and identified only by a meaningless coined term having no basis in the literature cannot constitute "such full, clear, concise and exact terms as to enable any person skilled in the art *** to make and use" a program allegedly performing functions, operations and interactions so numerous and complex as to require several pages of specification merely to list them. The mere words "Text Code" may be "concise", but as a disclosure it is neither "full", nor "clear" nor "exact", and therefore is not "enabling".

THE BELATED LAWRENCE AMENDMENT RELIED UPON IN THE FINAL REJECTION

A portion of the Lawrence et al. patent twice relied upon in the final rejection was a limitation inserted into Claim 1 by amendment about 1 1/2 years after the filing date to which appellant was deemed entitled. This limitation reads:

> "means in said editing means for invoking said formatting means when a text editing step is completed."

This limitation was inserted into the Lawrence et al. claim by an amendment filed Feb. 17, 1984. In Paragraph 11 of the Action mailed Sept. 18, 1987 the Examiner held that appellant's present application is entitled to the filing date of Sept. 28, 1982 of his co-pending parent application Serial No. 425,612. Appellant's entitled filing date therefore antedates by about 1 1/2 years the claim insertion repeatedly relied upon by the Examiner.

THE LAWRENCE INTERPRETER/FORMATTER IS NOT "INTERRUPTED"

The term "interrupted", as used in the computer art and appellant's specification and claims, has a different meaning than "terminated". The fact that the Lawrence et al. interpreter/formatter is described as "terminated" if a key is depressed during formatting is inconsistent with the Examiner's contention that the interpreter/formatter has been "interrupted" or that the interpreter/formatter continuously maintains control "except during interrupts". As explained in Paragraph 13 of the third Reiffin affidavit, a routine is said to be "terminated" when it has either completed its sequence of instructions or has aborted. In either event, the routine will not execute again unless and until it is invoked all over again from the beginning of its instruction sequence.

On the other hand, a routine is said to be "interrupted" when it temporarily loses control of the central processing unit to an interrupt service routine. When the latter is finished, the interrupted routine then automatically resumes its control of the central processing unit and continues executing its instructions at the very point in the instruction sequence where it was interrupted. Concisely stated, an "interrupted" routine is not "terminated" and a "terminated" routine is not "interrupted".

The term "interrupted" also describes a different operation than that implied by the terms "invoked" and "initiated". That the Lawrence specification states at Column 3, Lines 30-38 that the text editor 10 "invokes" and "re-invokes"

the interpreter/formatter 6, and at Column 3, Lines 25-29 that the latter is "initiated" by the editor, are statements inconsistent with an "interrupt" of the interpreter/formatter by the editor.

As explained in Paragraph 14 of the third Reiffin affidavit, in the computer art a routine (procedure or subprogram) is said to be "invoked" by another routine when it is "called" by the latter. The first (calling) routine invokes the second (invoked) routine by executing a "call" instruction. This stores the memory address of the second (invoked) routine into the program counter of the computer. The central processing unit then obeys the program counter and jumps to the address of the second (invoked) routine by executing the instructions stored at said address. That is, the first (calling) routine knows and determines the symbolic address in memory of the second (invoked) routine to which the central processing unit is to jump.

Such a call or invocation is different from an interrupt. In the interrupt operation the first routine has no knowledge of the second routine and has no control of the destination of the central processing unit or of when the jump to the second routine takes place. Insofar as the first routine is concerned, all it "knows" is that it has temporarily lost control of the central processing unit to some unknown interrupt service routine and that it will automatically regain its control when the interrupt service routine has finished.

The fact that the Lawrence interpreter/formatter is described as being "initiated", "invoked" and "re-invoked" is inconsistent with the Examiner's contention that it maintains

control of the system except during interrupts. As explained above, an interrupted routine does not resume control by being "initiated" or "invoked". Instead it resumes control automatically when the interrupt service routine terminates. Also, it resumes at the very point in its instruction sequence where it was interrupted, and no "initiation" occurs.

Furthermore, as alleged in Paragraph 16 of the third Reiffin affidavit, when a first routine is interrupted by a second routine, the latter gets control of the central processing unit immediately, and not after some other delayed subsequent event such as the interrupt/formatter reaching the end of the current row of the text being processed. Col. 11, Lines 51-54 of the Lawrence specification states:

> "If a key is depressed during formatting, then the interpreter is terminated at the end of the current row and control is passed back to the keystroke processor 9". (Emphasis added.)

This delay in passing control back to the keystroke processor makes it clear that the interpreter is not "interrupted" by the depression of a key. If the keystroke processor 9 and text editor 10 were embodied in an interrupt service routine, they would get control immediately upon activation of the interrupt rather than waiting until after the interpreter/formatter reaches "the end of the current row" of the text being formatted. This immediate acquisition of control is inherent and inevitable by the very definition and nature of an interrupt.

Most revealing of all is the fact that nowhere in the Lawrence et al. specification do they state that the interpreter/formatter is "interrupted". Only in Claim 4 of the

patent is there a confused self-contradictory statement that "the editing means includes means to interrupt the formatting means...invoking said editing means". The author of this limitation was not aware that the editing means cannot invoke anything until it gets control of the central processing unit, and that once it gets such control there would be no point in invoking itself. This claim, unlike the description in the specification, appears to have been drafted by a patent attorney unfamiliar with the interrupt mechanism.

THE LAWRENCE INTERPRETER/FORMATTER DOES NOT MAINTAIN CONTROL OF THE SYSTEM

The Examiner states in Paragraph 11 of the final rejection that:

> "During the formatting operation, the interpreter/formatter normally has control of the system."

This statement is either tautological or inaccurate. If taken literally, it is merely a tautology; that is, a statement which is inevitably true solely by virtue of its logical form, but which actually states nothing factual. For example, equally true and meaningless would be a statement such as:

> "During the segmenting operation, the segmenter normally has control of the system."

In any event, the interpreter/formatter does not "normally" have control of the system in the same sense that appellant's Claim 51 recites that his code processor program continuously maintains control except during interrupts. The Lawrence specification states that the interpreter/formatter gets

control only when it is repeatedly "invoked" or "re-invoked" by the text editor 10 or segmenter 5. As explained above and in Paragraph 14 of the third Reiffin affidavit, an interrupted routine does not regain control of the system by being "invoked". Instead it automatically resumes control when the interrupt service routine terminates.

THE LAWRENCE INTERRUPT LINE 29 DOES NOT DISCLOSE APPELLANT'S CLAIMED MODE OF OPERATION

The Lawrence specification (Col. 6, Lines 48-49) mentions "an interrupt line 29" shown in Fig. 5 as extending between the microprocessor 26 and keyboard adapter 30. As explained in Paragraph 19 of the third Reiffin affidavit, this does not disclose or suggest the mode of operation recited in appellant's claims. At very most, this might suggest to one skilled in the art the conventional operation of the Personal Computer of IBM, the assignee of the Lawrence et al. patent.

As shown in Exhibit A of the third Reiffin affidavit, which exhibit includes copies of Pages 92-101 of "8088 Assembly Language Programming The IBM PC" by D. C. Willen and J. L. Krantz, in the IBM Personal Computer the striking of a key activates an interrupt which stores a corresponding character code in a type-ahead buffer. The main program (such as the Lawrence interpreter/formatter 6) running at the time then immediately resumes control of the central processing unit.

No successor routine (such as Lawrence's text editor 10) takes control as an interrupt service routine. To change control from the main program to an editor it would be necessary for the

main program either to terminate or to invoke the editor. The former would appear to be what is intended by Lawrence et al. since it is consistent with their statement that the interpreter is "terminated" if a key is depressed.

Although it would be possible for a programmer to change the interrupt vectors stored in the IBM Personal Computer so that the interrupt service routine includes an editor which is directly and immediately given control by the interrupt, this modification is not the conventional mode of operation and is not disclosed in the Lawrence et al. patent. There is nothing in their disclosure to suggest that activation of their interrupt line 29 does anymore than the conventional keystroke operation in the IBM Personal Computer.

LAWRENCE DOES NOT DISCLOSE PASSING CONTROL BACK TO THE INTERPRETER/FORMATTER AFTER EACH KEYSTROKE

Paragraph 11 of the final rejection cites the Lawrence et al. claim limitation at Column 14, Lines 15-17 for the alleged "interaction" characterized by the Examiner as follows:

> "After the editor has completed the entry of the code byte, the system control is returned to the interpreter/formatter." (Emphasis added.)

However, the cited claim limitation does not say this. Instead it states:

> "means in said editing means for invoking said formatting means when a text editing step is completed." (Emphasis added.)

The difference between the Examiner's phrase "the code byte" and the actual phrase "text editing step" is critical. The

Examiner's misread version would tend to support his erroneous contention that the reference patent discloses that

> "control of the system is passed back to the interpreter/formatter from the editor after each of the keystroke [sic]." (Emphasis added.)

Appellant can find no such disclosure in the reference. Instead it appears that control is not passed back to the interpreter/formatter until completion of a "text editing step" which usually comprises a plurality of keystrokes. Lawrence et al. refer to this as "completion of an update" in Column 3, Line 29. Nowhere in the reference patent can appellant find any statement that system control reverts to the interpreter/formatter after each keystroke or after each entry of a code byte.

The Examiner's interpretation that control is passed back to the interpreter/formatter after each keystroke is also allegedly supported by the limitation "keystroke-by-keystroke basis" in Column 13, Line 29 of Lawrence's Claim 1. However, the context of this limitation reveals that the Examiner's interpretation is unfounded. This limitation refers only to the "editing means" and has no relation to the "formatting means". The latter is not even alluded to until the last paragraph of the claim where it is recited that the formatting means is invoked "when a text editing step has been completed." As discussed above, since a text editing step usually comprises a plurality of keystrokes, there is no disclosure in the reference that control is passed back after each keystroke.

Furthermore, there would be no reason or purpose for passing control back to the interpreter/formatter after each keystroke and before the plurality of keystrokes comprising an editing operation is completed. This pointless mode of operation is even more unlikely in view of the possible need to segment the text after the editing operation and before the formatting operation. As alleged in Paragraph 22 of the third Reiffin affidavit, a screen formatter, such as the Lawrence interpreter/formatter purports to be, takes only a fraction of a second to format a screenful of text. Therefore it would be pointless to reformat the entire screen repeatedly for each keystroke and thereby reformat the screen numerous times for each complete editing operation. On the other hand, in appellant's system a compilation may take a very long time and hence it is vital that processing be done between keystrokes to minimize the tedious waiting periods inherent in conventional compilation methods.

APPEALED CLAIMS 51 AND 54 RECITE FURTHER DISTINGUISHING LIMITATIONS NOT NOTED IN THE FINAL REJECTION

Claims 51 and 54 were entered after final rejection and recite further distinguishing limitations beyond the limitations noted in the final rejection. In the following copy of Claim 51 appellant has underlined the more important limitations which he relies upon as distinguishing his invention over the Lawrence et al. reference:

51. A computer system for the concurrent entry and processing of code, and comprising

a console having keys for entry of successive characters constituting said code,

a memory storing a buffer for holding said code, a code processor program consisting of a sequence of machine instructions, an interrupt service routine at a predetermined memory address, and a stack,

a central processing unit for executing said code processor program instructions and having an interrupt input, a program counter for storing the memory address of the next instruction to be executed by the central processing unit, and means immediately responsive to activation of the interrupt input to push the contents of the program counter onto said stack and to load into said program counter said memory address of the interrupt service routine,

means to cause the central processing unit to execute said code processor program instructions whereby the code processor program thereafter continuously maintains control of the central processing unit, except during interrupts, so as to process the code concurrently as the code is being entered into the system,

means responsive to striking one of said keys to activate said interrupt input whereby the contents of the program counter are pushed onto the stack and the memory address of the interrupt service routine is loaded into the program counter thereby immediately passing control of the central processing unit to the interrrupt service routine, whereupon the code processor program is interrupted immediately upon completing the machine instruction which it was executing when the key was struck,

said interrupt service routine including an editor program for inserting into said buffer a character code corresponding to said struck key, and having a return instruction at the end thereof,

said central processing unit having means responsive to execution of said return instruction to pop from the stack into the program counter the memory address of the code processor program instruction immediately following the instruction interrupted by striking said key, whereby the code processor program continues processing said character codes at the location in the buffer where the code processor program was interrupted.

The Lawrence et al. reference does not disclose:

(1) inclusion of the editor in an interrupt service routine;

(2) a return instruction at the end of the interrupt service routine;

(3) means responsive to striking a key to load the memory address of the interrupt service routine into the program counter thereby immediately interrupting the code processor program and immediately passing control of the central processing unit to the interrupt service routine; and

(4) means responsive to execution of the return instruction to pop from the stack into the program counter the memory address of the code processor program instruction immediately following the interrupted instruction;

(5) whereby the code processor program continues processing character codes in the buffer at the point where it was interrupted.

Claim 54 recites several of these distinguishing aspects in somewhat broader terms.

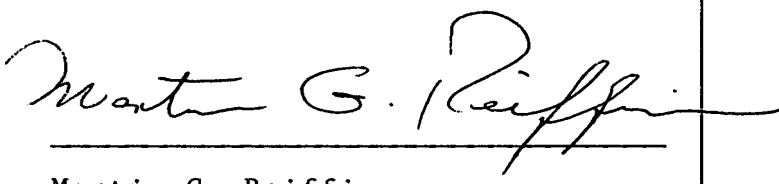
Claim 56 is still broader than the above-discussed claims and is urged as allowable over the reference on the ground that the functions recited in this claim are not enabled in the reference.

CONCLUSION

It is respectfully submitted that Claims 51, 54 and 56 are patentable and that the Examiner's refusal to allow these claims should be reversed.

Appellant does not request an oral hearing.

Respectfully submitted,



August 2, 1988

Martin G. Reiffin,
Appellant
3895 Cottonwood Drive
Danville, CA 94526
(415) 838-6980